Your search matched **13** of **976857** documents.

A maximum of **13** results are displayed, **15** to a page, sorted by **Relevance** in **descending** order.
You may refine your search by editing the current search expression or entering a new one the text box.

Then click **Search Again**.

(single instruction)and (loop)

Search Again

Results:
Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD**

1 **A topological sorting and loop cleansing algorithm for a constrained MIMD compiler of shift-invariant flow graphs**
Lee, S.; Barnwell, T., III;
Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86. , Volume: 11 , Apr 1986
Page(s): 2927 -2930

[Abstract] [PDF Full-Text (176 KB)] **IEEE CNF**

2 **Structure-based automatic extraction of the program heterogeneity**
Guosun Zeng; Xinda Lu; Jingcun Wang; Dingkang Zhou;
High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on , Volume: 1 , 14-17 May 2000
Page(s): 261 -262 vol.1

[Abstract] [PDF Full-Text (140 KB)] **IEEE CNF**

3 **Fast software implementation of MPEG advanced audio encoder**
Dimkoviae, I.; Milovanoviae, D.; Bojkoviae, Z.;
Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on , Volume: 2 , 1-3 July 2002
Page(s): 839 -843 vol.2

[Abstract] [PDF Full-Text (421 KB)] **IEEE CNF**

4 **Experimental performance evaluation of the clustered multiprocessor system MUGEN**
Horiguchi, S.; Kawazoe, Y.;
TENCON '89. Fourth IEEE Region 10 International Conference , 22-24 Nov. 1989
Page(s): 205 -208

[Abstract] [PDF Full-Text (264 KB)] **IEEE CNF**

5 **Data parallel computers and the FORALL statement**
Albert, E.; Lukas, J.D.; Steele, G.L., Jr.;
Frontiers of Massively Parallel Computation, 1990. Proceedings., 3rd Symposium on the , 8-10 Oct. 1990
Page(s): 390 -396

[Abstract] [PDF Full-Text (480 KB)] **IEEE CNF**

6 **Compiling SIMD programs for MIMD architectures**
Quinn, M.J.; Hatcher, P.J.;
Computer Languages, 1990., International Conference on , 12-15 March 1990
Page(s): 291 -296

[Abstract] [PDF Full-Text (440 KB)] **IEEE CNF**

7 **Fast barrier synchronization hardware**
Beckmann, C.J.; Polychronopoulos, C.D.;
Supercomputing '90. Proceedings of , 12-16 Nov. 1990
Page(s): 180 -189

[Abstract] [PDF Full-Text (676 KB)] **IEEE CNF**

8 **A decoupled access/execute processor for matrix algorithms: architecture and programming**
Moreno, J.H.; Figueroa, M.E.;
Application Specific Array Processors, 1991. Proceedings of the International Conference on , 2-4 Sept. 1991
Page(s): 281 -295

[Abstract] [PDF Full-Text (568 KB)] **IEEE CNF**

9 **FORGE 90: a parallel programming environment**
Levesque, J.M.;
Compcon Spring '92. Thirty-Seventh IEEE Computer Society

International Conference, Digest of Papers. , 24-28 Feb. 1992
Page(s): 291 -294

[Abstract] [PDF Full-Text (288 KB)] **IEEE CNF**

---

10 **A global mode instruction minimization technique for embedded DSPs**
Wilson, T.C.; Grewal, G.W.;
VLSI, 1996. Proceedings., Sixth Great Lakes Symposium on , 22-23 March 1996
Page(s): 18 -21

[Abstract] [PDF Full-Text (304 KB)] **IEEE CNF**

---

11 **Extracting SIMD parallelism from `for' loops**
Gustin, V.; Bulic, P.;
Parallel Processing Workshops, 2001. International Conference on , 3-7 Sept. 2001
Page(s): 23 -28

[Abstract] [PDF Full-Text (360 KB)] **IEEE CNF**

---

12 **Exploiting operation level parallelism through dynamically reconfigurable datapaths**
Zhining Huang; Sharad, M.;
Design Automation Conference, 2002. Proceedings. 39th , 10-14 June 2002
Page(s): 337 -342

[Abstract] [PDF Full-Text (799 KB)] **IEEE CNF**

---

13 **Extending value reuse to basic blocks with compiler support**

Huang, J.; Lilja, D.J.;
Computers, IEEE Transactions on , Volume: 49 Issue: 4 , April 2000
Page(s): 331 -347

[Abstract] [PDF Full-Text (7232 KB)] **IEEE JNL**

CiteSeer Find: [ ] Documents Citations

Searching for **PHRASE pipeline loop**.
Restrict to: Header Title Order by: Citations Hubs Usage Date Try: Amazon B&N Google (RI) Google (Web) CSB DBLP
18 documents found. **Order: citations weighted by year.**

Sentinel Scheduling for VLIW and Superscalar Processors - Mahlke (1992) (Correct) (36 citations)
can be used in conjunction with software **pipeline loop** scheduling [4] or straight-line code
ftp.crhc.uiuc.edu/pub/IMPACT/conference/asplos-92-sentinel.ps

Using Iterative Compilation for Managing Software.. - van der Mark, Rohou, al. (1999) (Correct) (5 citations)
Our Work. 2. Loop Unrolling And Software **Pipeline Loop** Unrolling And Software Pipeline Are Two Major
www.liacs.nl/~pmark/publications/scopes99.ps

Software Pipelining with Register Allocation and Spilling - Wang, Krall, Ertl, Eisenbeis (1994) (Correct) (11 citations)
4. t2=s*s 5. t3=t1*t2 6. a[t0]t3 (1) The **Loop Pipeline** Number Operation Latency Memory port 2 Load
ftp.inria.fr/INRIA/Projects/a3/eisenbei/micro27.ps.Z

Sentinel Scheduling with Recovery Blocks - David August Brian (1995) (Correct) (4 citations)
execution -used in conjunction with software **pipeline loop** scheduling [7] or straight-line code
www.crhc.uiuc.edu/IMPACT/ftp/report/crhc-95-05.sentinel.ps.Z

Memory Access Optimization and RAM Inference for Pipeline.. - Weinhardt, Luk (1999) (Correct) (1 citation)
during the iterations of the outer while **loop. Pipeline** vectorization consists of three main steps:
www.doc.ic.ac.uk/~mw8/papers/fpl99.pdf.gz

Mapping Loops on Coarse-Grain Reconfigurable Architectures.. - Lee, Choi, Dutt (Correct)
executions using a novel organization of the **loop pipeline**. We develop the conditions for sharing memory
number of lines used for one instance of the **loop pipeline**, b) the number of configurations, c) the
www.cecs.uci.edu/technical_report/TR02-34.pdf

Acceleration of First and Higher Order Recurrences on - Processors With Instruction (Correct)
Keywords recurrences, parallelism, software **pipeline, loop** optimization, height reduction,
www.hpl.hp.com/research/itc/car/papers/../papers/Acceleration.pdf

Height Reduction of Control Recurrences for ILP Processors - Michael Schlansker Vinod (Correct)
blocked back-substitution, software **pipeline, loop** optimization 1 Introduction Control and
www.hpl.hp.com/research/itc/car/papers/../papers/Control-height.pdf

Parallelization of Control Recurrences for ILP Processors - Michael Schlansker Vinod (Correct)
blocked back-substitution, software **pipeline, loop** optimization 1 Introduction Control and
www.hpl.hp.com/research/itc/car/papers/../papers/parallelization.pdf

Building Parallel Applications using Design Patterns - Goswami, Singh, Preiss (2000) (Correct)
supports patterns supporting replication, **pipeline, loop** and conditional constructs. Tracs is another
www.pads.uwaterloo.ca/Bruno.Preiss/papers/published/2000/cser/paper.ps

Memory Access Optimization for Reconfigurable Systems - Weinhardt, Luk (Correct)
the innermost for loop (line 8-19) is such a **pipeline loop**. It contains the functions Fmin and Fmax (not
xN2 x8 for (int y=0 yN2 ypipeline loop *9 if (xN &yN) 10 im_out[x,y] y2
www.markus-weinhardt.de/papers/memopt.ps.gz

Custom Embedded Counterflow Pipelines - Childers (2000) (Correct)
instruction set architecture for the software **pipeline loop**. As part of this work, techniques were
Figure 50: Software **pipeline loop**.
to match the dynamic behavior of a kernel **loop. Pipeline** refinement is very simple: it identifies
ftp.cs.virginia.edu/pub/dissertations/2000-06.ps.Z

Dynamic Branch Decoupled Architecture - Akhilesh Tyagi Hon-Chi (Correct)

Search Results

Search Results for: **[pipeline<AND>((SIMD instruction<AND>((loop<AND>((single instruction)))))]**
Found **18** of **121,820 searched.**

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 18 of 18 short listing

1 Research sessions: implementation techniques: Implementing database operations using SIMD instructions 85%
Jingren Zhou , Kenneth A. Ross
Proceedings of the 2002 ACM SIGMOD international conference on Management of data June 2002
Modern CPUs have instructions that allow basic operations to be performed on several data elements in parallel. These instructions are called SIMD instructions, since they apply a single instruction to multiple data elements. SIMD technology was initially built into commodity processors in order to accelerate the performance of multimedia applications. SIMD instructions provide new opportunities for database engine design and implementation. We study various kinds of operations in a database con ...

2 MOM: a matrix SIMD instruction set architecture for multimedia applications 82%
Jesus Corbal , Roger Espasa , Mateo Valero
Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM) January 1999

3 Integrating SIMD into the undergraduate curriculum 82%
W. D. Maurer
The Journal of Computing in Small Colleges , Proceedings of the sixth annual CCSC northeastern conference on The journal of computing in small colleges April 2001
Volume 16 Issue 4

Assembly language instruction today, in our view, should include instruction in the newly important area of single-instruction, multiple-data (SIMD) instructions. Such instructions are available on all major platforms, and they considerably speed up operations on arrays, particularly large arrays. This speedup is more pronounced with assembly language than with

algebraic language programming, and thus provides another reason for undergraduate students to learn assembly language. We discuss th ...

4 Simulation and architecture evaluation: Vector vs. superscalar and VLIW architectures for embedded multimedia benchmarks 80%

Christoforos Kozyrakis , David Patterson

Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture November 2002

Multimedia processing on embedded devices requires an architecture that leads to high performance, low power consumption, reduced design complexity, and small code size. In this paper, we use EEMBC, an industrial benchmark suite, to compare the VIRAM vector architecture to superscalar and VLIW processors for embedded multimedia applications. The comparison covers the VIRAM instruction set, vectorizing compiler, and the prototype chip that integrates a vector processor with DRAM main memory.We de ...

5 PACT 2001 workshops: MediaBreeze: a decoupled architecture for accelerating multimedia applications 80%

Deependra Talla , Lizy K. John

ACM SIGARCH Computer Architecture News December 2001

Volume 29 Issue 5

Decoupled architectures are fine-grain processors that partition the memory access and execute functions in a computer program and exploit the parallelism between the two functions. Although some concepts from the traditional decoupled access execute paradigm made its way into commercial processors, they encountered resistance in general-purpose applications because these applications are not very structured and regular. However, multimedia applications have recently become dominant workload on ...

6 Exploiting SIMD parallelism in DSP and multimedia algorithms using the AltiVec technology 80%

Huy Nguyen , Lizy Kurian John

Proceedings of the 13th international conference on Supercomputing May 1999

7 Exploiting instruction level parallelism in geometry processing for three dimensional graphics applications 80%

Chia-Lin Yang , Barton Sano , Alvin R. Lebeck

Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture November 1998

8 Growing discord: programming philosophy and hardware design 77%

K. W. Neves

Proceedings of the 1988 ACM/IEEE conference on Supercomputing November 1988

Generally, vector compiler technology has been successful in achieving reasonable peak efficiency on “good” code. Moreover, the community's ability to generate “good” vector code has improved dramatically. As we move into the era of parallelism, particularly in supercomputing, we can observe certain trends among the leaders in compiler technology. The basic techniques are extensions of strategies for vector machines, but have limited effectiveness in a parallel envir ...

9 A survey of processors with explicit multithreading 77%
Theo Ungerer , Borut Robi? , Jurij Šilc
ACM Computing Surveys (CSUR) March 2003
Volume 35 Issue 1
Hardware multithreading is becoming a generally applied technique in the next generation of microprocessors. Several multithreaded processors are announced by industry or already into production in the areas of high-performance microprocessors, media, and network processors.A multithreaded processor is able to pursue two or more threads of control in parallel within the processor pipeline. The contexts of two or more threads of control are often stored in separate on-chip register sets. Unused i ...

10 Articles: Blurring Lines Between Hardware and Software 77%
Homayoun Shahri
Queue April 2003
Volume 1 Issue 2

11 Static resource models for code-size efficient embedded processors 77%
Qin Zhao , Bart Mesman , Twan Basten
ACM Transactions on Embedded Computing Systems (TECS) May 2003
Volume 2 Issue 2
Due to an increasing need for flexibility, embedded systems embody more and more programmable processors as their core components. Due to silicon area and power considerations, the corresponding instruction sets are often highly encoded to minimize code size for given performance requirements. This has hampered the development of robust optimizing compilers because the resulting irregular instruction set architectures are far from convenient compiler targets. Among other considerations, they int ...

12 Ray tracing on programmable graphics hardware 77%
Timothy J. Purcell , Ian Buck , William R. Mark , Pat Hanrahan
ACM Transactions on Graphics (TOG) , Proceedings of the 29th annual conference on Computer graphics and interactive techniques July 2002
Volume 21 Issue 3
Recently a breakthrough has occurred in graphics hardware: fixed function pipelines have been replaced with programmable vertex and fragment processors. In the near future, the graphics pipeline is likely to evolve into a general programmable stream processor capable of more than simply feed-forward triangle rendering.In this paper, we evaluate these trends in programmability of the graphics pipeline and explain how ray tracing can be mapped to graphics hardware. Using our simulator, we analyze ...

13 Energy aware compilation for DSPs with SIMD instructions 77%
Markus Lorenz , Lars Wehmeyer , Thorsten Dräger
ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems June 2002
Volume 37 Issue 7
The growing use of digital signal processors (DSPs) in embedded systems necessitates the use of optimizing compilers supporting special hardware features. In this paper we present compiler optimizations with the aim of minimizing energy consumption of embedded

applications: This comprises loop optimizations for exploitation of SIMD instructions and zero overhead hardware loops in order to increase performance and decrease the energy consumption. In addition, we use a phase coupled code generator ...

14 Embedded tutorial: Code generation for embedded processors 77%

Rainer Leupers

Proceedings of the 13th international symposium on System synthesis September 2000

The increasing use of programmable processors as IP blocks in embedded system design creates a need for C/C++ compilers capable of generating efficient machine code. Many of today's compilers for embedded processors suffer from insufficient code quality in terms of code size and performance. This violates the tight chip area and real-time constraints often imposed on embedded systems. The reason is that embedded processors typically show architectural features which are not well handled by class ...

15 Polygon rendering on a stream architecture 77%

John D. Owens , William J. Dally , Ujval J. Kapasi , Scott Rixner , Peter Mattson , Ben Mowery

Proceedings 2000 SIGGRAPH/EUROGRAPHICS workshop on on Graphics hardware August 2000

The use of a programmable stream architecture in polygon rendering provides a powerful mechanism to address the high performance needs of today's complex scenes as well as the need for flexibility and programmability in the polygon rendering pipeline. We describe how a polygon rendering pipeline maps into data streams and kernels that operate on streams, and how this mapping is used to implement the polgyon rendering pipeline on Imagine, a programmable stream processor. We compare our resul ...

16 Exploiting a new level of DLP in multimedia applications 77%

Jesus Corbal , Mateo Valero , Roger Espasa

Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture November 1999

This paper proposes and evaluates MOM: a novel ISA paradigm targeted at multimedia applications. By fusing conventional vector ISA approaches together with more recent SIMD-like (Single Instruction Multiple Data) ISAs (such as MMX), we have developed a new matrix oriented ISA which efficiently deals with the small matrix structures typically found in multimedia applications. MOM exploits a level of DLP not reachable by neither conventional vector ISAs nor SIMD-like media ISA extensi ...

17 Optimizing the data cache performance of a software MPEG-2 video decoder 77%

Peter Soderquist , Miriam Leeser

Proceedings of the fifth ACM international conference on Multimedia November 1997

18 PixelFlow: the realization 77%

John Eyles , Steven Molnar , John Poulton , Trey Greer , Anselmo Lastra , Nick England , Lee Westover

Proceedings of the 1997 SIGGRAPH/Eurographics workshop on Graphics hardware August 1997

Results 1 - 18 of 18 short listing

CiteSeer Find: single instruction and loop and pipelir Documents Citations

Searching for **single instruction and loop and pipeline**.
Restrict to: Header Title Order by: Citations Hubs Usage Date Try: Amazon B&N Google (RI) Google (Web) CSB DBLP
45 documents found. **Order: citations weighted by year.**

The Superthreaded Architecture: Thread Pipelining with Run-time.. - Tsai, Yew (1996) (Correct) (57 citations)
blocks need to be grouped together in a **single instruction** stream. As a larger instruction window size
the superthreaded architectural model can exploit **loop**-level parallelism from a broad range of
serious when a compiler attempts to software **pipeline** a **loop** with conditional branches [18]In
www-users.cs.umn.edu/Research/Agassiz/Paper/tsai.pact96.ps.Z

One or more of the query terms is very common - only partial results have been returned. Try Google (RI).

IMPACT: An Architectural Framework for.. - Chang, Mahlke.. (1991) (Correct) (91 citations)
achieve solid speedup over high-performance **single-instruction**-issue processors. We ran experiments to
function inline expansion, instruction placement, **loop** unrolling, **loop** peeling, memory disambiguation,
[Kogge 81]By optimizing a simple instruction **pipeline** structure, current **pipelined** processors can
ftp.crhc.uiuc.edu/pub/IMPACT/conference/isca-91-framework.ps

Efficient Microarchitecture Modeling and Path Analysis for.. - Li, Malik, Wolfe (1996) (Correct) (32 citations)
timing analysis. The execution time of a **single instruction** depends on many factors and varies more
absence of dynamic structures and (iii) bounded **loops**. These restrictions can be imposed either through
for modern processors due to the presence of **pipelined** instruction execution units and cached memory
ftp.ee.princeton.edu/pub/yauli/rtss95.ps.gz

Compiling Fortran D for MIMD Distributed-Memory Machines - Hiranandani (1992) (Correct) (47 citations)
Parallel Computer Forum (PCF) Fortran [24]**Single-instruction**, multiple-data (SIMD) machines such as the
with explicit synchronization and parallel **loops** found in Parallel Computer Forum (PCF) Fortran
www.cs.umd.edu/~keleher/papers/fortrand.ps.gz

Zero-Cycle Loads: Microarchitecture Support for Reducing Load.. - Austin (1995) (Correct) (25 citations)
levels of the data memory hierarchy in a **single instruction**. A significant body of work is dedicated to
eliminates the entire load operation, or (**loop**) blocking eliminates many cache miss latencies. In
up to two cycles earlier than traditional **pipeline** designs. For a **pipeline** with one cycle data
ftp.cs.wisc.edu/sohi/papers/1995/micro.zcl.ps.gz

The MMachine Multicomputer - Fillo, Keckler, Dally, al. (1995) (Correct) (17 citations)
one for each ALU. All operations in a **single instruction** issue together but may complete out of
parallelized byidentifying tasks, such as **loop** iterations, that can be distributed both across
year #10#As a result, a 64-bit processor with a **pipelined** FPU #400M# 2 is only 11# of a 3.6G# 2 1993
publications.ai.mit.edu/ai-publications/pdf/AIM-1532.pdf

The M--Machine Multicomputer - Marco Fillo (1995) (Correct) (17 citations)
one for each ALU. All operations in a **single instruction** issue together but may complete out of
parallelized by identifying tasks, such as **loop** iterations, that can be distributed both across
year [10]As a result, a 64-bit processor with a **pipelined** FPU (400M 2 is only 11% of a 3.6G 2 1993
cva.stanford.edu/pub/publications/AI1532.ps.Z

The Superthreaded Processor Architecture - Jenn-Yuan Tsai (1999) (Correct) (5 citations)
from different basic blocks in a **single instruction** stream need to be examined and issued
serious when a compiler attempts to **pipeline** a **loop** with many conditional branches [21]In
is especially serious when a compiler attempts to **pipeline** a **loop** with many conditional branches [21]In
www.cs.umn.edu/Research/Agassiz/Paper/tsai.ieee.ps.gz

Cost-effective Hardware Acceleration of Multimedia Applications - Deependra Talla And (2001) (Correct) (1 citation)
processors (GPPs) have been enhanced with **single instruction** multiple data (SIMD) execution units [1]
packing/unpacking, permute, loads/stores, and **loop** branches) dominate media instruction streams

in determining the maximum clock rate? How many **pipeline** stages does the hardware add to the processor
www.ece.utexas.edu/projects/ece/lca/ps/deepu-iccd01.pdf

Caching and Predicting Branch Sequences for Improved Fetch.. - Önder, Xu, Gupta (1999) (Correct) (2 citations)
two as their execution is separated by a **single instruction** that performs a compare (c =EOLN)
in Figure 1, assume that during the execution of a **loop** iteration both the conditionals evaluate to false,
for speculative execution in the processor **pipeline**. Thus, if the predictions are correct the
www.cs.arizona.edu/people/gupta/research/Publications/Comp/pact99.ps

Performance Nonmonotonicities: A Case Study of the UltraSPARC.. - Kushman (1998) (Correct) (3 citations)
exist in which the addition or removal of a **single instruction** changes the performance of a program by a
in 6 cycles per iteration. 29 3-13 Assembly code **loop** which is executed at the maximum execution rate of
3-1. 20 3-3 The nine-stage **pipeline** of the UltraSPARC.
www.lcs.mit.edu/publications/pubs/pdf/MIT-LCS-TR-782.pdf

Comparing Software and Hardware Schemes For Reducing the.. - Hwu, Conte, Chang (1989) (Correct) (16 citations)
compare instructions [8]9]A case for **single-instruction** conditional branches is given in [6]When
that backward branches are usually at the end of **loops**. In the study done by J. E. Smith [4]the
disrupt the flow of instructions through the the **pipeline**, increasing the overall execution cost of branch
ftp.crhc.uiuc.edu/pub/IMPACT/conference/isca-89-branch.ps

Improving Software Pipelining With Unroll-and-Jam - Carr, Ding, Sweany (1996) (Correct) (5 citations)
that in our simple machine an add requires a **single instruction** to produce a result while the multiplier
been developed [1, 2, 3, 4]Unfortunately, not all **loops** have enough parallelism in the innermost **loop**
www.cs.rice.edu/~cding/documents/unroll.ps.gz

Vector Instruction Set Support for Conditional Operations - Smith Greg Faanes (2000) (Correct) (1 citation)
(typically several 10s to 100s) in a **single instruction**. These instructions are executed in a
Support for conditional operations (as occur in **loops** containing IF statements) is an important aspect
with SIMD implementations, but long vector, **pipelined** implementations have a number of advantages and
www.ece.wisc.edu/~jes/papers/isca00.smith.ps

Using Measurements to Derive the Worst-Case Execution Time - Lindgren, Hansson, Thane (2000) (Correct)
(1 citation)
machine timing effects that depend on a **single instruction** and its immediate neighbors. Examples of
flow analysis (such as number of iterations in **loops**) with low-level timing information. This paper
show how it can be extended to architectures with **pipelines**. 1. Introduction Worst-case execution time
www.mrtc.mdh.se/publications/0257.ps

Sentinel Scheduling with Recovery Blocks - David August Brian (1995) (Correct) (4 citations)
files as the maximum number of branches any **single instruction** is speculated above. Given that some
- used in conjunction with software **pipeline loop** scheduling [7] or straight-line code scheduling
execution -used in conjunction with software **pipeline loop** scheduling [7] or straight-line code
www.crhc.uiuc.edu/IMPACT/ftp/report/crhc-95-05.sentinel.ps.Z

An Implementation Of Gurpr*: A Software Pipelining Algorithm - Bockhaus (1992) (Correct) (2 citations)
: 2 2.1: A **Single Instruction** (Node) in the EPS Machine Model.
graph :17 3.2.4 Pipelining the **loop** body :19
www.crhc.uiuc.edu/IMPACT/ftp/report/ms-thesis-john-bockhaus.ps.Z

Program Balance and its Impact on High Performance RISC.. - Lizy Kurian (1995) (Correct) (1 citation)
to the importance of memory bandwidth. **Single instruction** stream parallelism will not be much
defined two metrics called `machine balance' and `**loop** balance' which together indicate how efficiently a
Access/Execute Balance, Memory Bandwidth, **Pipeline** Balance, Program Behavior. 1 Introduction
www.ece.utexas.edu/~ljohn/raleigh.ps

First 20 documents Next 20

Try your query at: Amazon Barnes & Noble Google (RI) Google (Web) CSB DBLP

CiteSeer Find: single instruction and loop and pipelir Documents Citations

Searching for **single instruction and loop and pipeline**.
Restrict to: Header Title Order by: Citations Hubs Usage Date Try: Amazon B&N Google (RI) Google (Web) CSB DBLP
45 documents found. **Order: citations weighted by year.**

Parallel Processing for Volume Visualization - Silva (1992) (Correct) (1 citation)
this paper. Of interest to our work are the **single-instruction** stream, multiple-data stream (SIMD) and the
to a SIMD algorithm by replacing each inner **loop** with a single broadcast instruction that
is to avoid binary decision during the rendering **pipeline**. Previous techniques for displaying surfaces
www.cs.sunysb.edu/~csilva/papers/claudio-rpe.ps

Experiments with Parallel Algorithms for Combinatorial.. - Kindervater, Trienekens (1985) (Correct) (1 citation)
different data at the same time. In SIMD (**single instruction** stream, multiple data stream) computers all
executed in parallel. For example the body of a for **loop** (which can be executed in parallel for each value
dataflow machine and the CDC-CYBER-205 (a **pipeline** computer)1980 Mathematics Subject
ftp.cs.few.eur.nl/pub/doc/webdoc/EUR-ECT-85-50/report.ps

Segment-Wise Timing and Power Measurement in Software Emulation - Wolf, Kruse, Ernst (Correct)
using a host trace and cost tables for the **single instructions** [9]While this first approach is fast but
instruction under investigation is executed in a **loop** instead of the real program. Power consumption of
to be a conservative selection with respect to **pipeline** and cache behavior as well as the global
www.ida.ing.tu-bs.de/research/publications/ps/WKE01:SegmeWiseTiminPower.ps.gz

Hardware Support to Reduce Overhead in Fine-Grain Media Codes - Talla, John, Burger (Correct)
and Motorola's AltiVec [1]2]Such SIMD (**single instruction** multiple data) extensions have been
this paper, we propose and evaluate a programmable **loop** engine (PLE for short) that executes media codes
www.ece.utexas.edu/projects/ece/lca/ps/LCA-TR-011101.ps

Performance Characterization of Intel's Internet Streaming SIMD.. - Godbole (2000) (Correct)
media-processing algorithms through **Single Instruction** Multiple Data (SIMD) techniques. Media
38 4.1.1 Simple **Loop** with Single Array
to the pixel- processing part of the 3D-geometry **pipeline**, where no floating-point computation is
www.ece.utexas.edu/projects/ece/lca/ps/godbole_report.pdf

Power-Aware Modulo Scheduling for High-Performance VLIW - Yun (Correct)
average power. In VLIW processors where a **single instruction** may contain a variable number of
OVERVIEW Software pipelining is an aggressive **loop** scheduling technique for VLIW processors. It
a VLIW machine model with a MIPS-like integer **pipeline** and an UltraSPARClike floating-point (FP) unit
davinci.snu.ac.kr/Download/step.pdf

Emmerald: A Fast Matrix-Matrix Multiply Using Intel's SSE.. - Aberdeen, Baxter (Correct)
for large matrices using the Intel Pentium **single instruction** multiple data (SIMD) floating point
efficiency. for (x =0 x m xouter **loop** *for (y =0 y n yfor (z =0 z k
of execution in the FPU (floating point unit) **pipeline** during any given cycle. The use of special
csl.anu.edu.au/~daa/files/emmerald.ps.gz

Using a Swap Instruction to Coalesce Loads and Stores - Apan Qasem David (Correct)
to coalesce loads and stores into a **single instruction**, which results in a reduction of memory
for a memory reference across iterations of a **loop**, unrolling the **loop**, and rescheduling instructions
requires the same number of cycles in the **pipeline** as a store instruction on the MicroSPARC I [10]
www.cs.fsu.edu/research/reports/TR-010501.ps

Dynamic Branch Decoupled Architecture - Akhilesh Tyagi Hon-Chi (Correct)
decoupling is a technique to decouple a **single instruction** stream program into two streams. One stream
The average speedup over 12 Lawrence Livermore **loops** was 1.58 with several **loops** achieving a speedup
frequency of about 20%the impact of branches on **pipeline** performance is quite significant. In this paper,
www.cse.msu.edu/rgroups/isal/pubs/conf/./dynamicarch.ps.gz

Performance of the iPSC/860 Node Architecture - Steven Moyer Ipc-Tr- (Correct)
to be fetched from the data cache with a **single instruction**, given proper data alignment this can be exploited to increase the performance of inner-**loop** computations it is not intended to provide a data paths, on chip data and instruction caches, **pipelined** floating-point arithmetic units and bus
ftp.cs.virginia.edu/pub/techreports/IPC-91-07.ps.Z

Future Branches - Beyond Speculative Execution - Appelbe, Das, Harmon (1997) (Correct)
address (where the branch will go to) in a **single instruction**. Future branches allows a branch condition well before we reach them. In a classic for **loop** a programmer can easily determine whether the next reaches the branch source, then there are no **pipeline** stalls or prediction necessary. Benchmark
ftp.cc.gatech.edu/pub/coc/tech_reports/1997/GIT-CC-97-33.ps.Z

A Fixed-Point DSP (MDSP) Chip for Portable Multimedia - Ong, Sunwoo (1999) (Correct)
chipscomxA8H emx y acom bination of SIMD (**Single Instruction**-stream Multiple Data-streamX superscalar, generation, instruction decoding, hardware FOR **loop** control, and e ception processing. To support up architecture shown in Fig. 2em7H ys a two-stage **pipeline** structure. The first stage is the input register
search.ieice.org/1999/files/../pdf/e82-a_6_939.pdf

A Performance Evaluation Of Multimedia Kernels Using.. - Julien Sebot Sebot (Correct)
AltiVec and G4 Architecture AltiVec is a **Single Instruction** Multiple Data (SIMD) machine comprised of: been hand tuned using well-known techniques like **loop** unrolling and software pipelining. For our The final speedup for the 3D geometry **pipeline** is 4 when using two dynamics light sources i.e.
iacoma.cs.uiuc.edu/caecw00/SUB/sebot.ps.Z

Assignment 2: SPARC ISA Performance Analysis and Pipeline.. - Unknown (1999) (Correct)
gathering and analysis of the execution of a **single instruction** -the next instruction to be executed in the shade_trctl_it routine)Then a typical inner **loop** is coded using the routine shade_step, with each Assignment 2: Sparc Isa Performance Analysis And **Pipeline** Simulation Due Date: In Lab On The Week Of
www.cse.nd.edu/class_data/cse322/www/../www/project1_99_shade.ps

Automatic Exploitation Of Concurrency in C: Is It Really So Hard? - Grob (1988) (Correct)
whereas in SIMD designs there is just a **Single Instruction** stream for the Multiple Data streams. with two arrays as the operands. Instead of a **loop** iterating through all the elements of the arrays, of the vector operation are passed into a **pipeline** of processors. Each processor performs a part of
ftp.nyu.edu/pub/ultra/ucn/101-150/ucn140.ps.Z

Theory of Modulo-Scheduled Pipelines - Govindarajan, Altman, al. (Correct)
is to support multiple versions of a **single instruction** i.e.different versions requiring a **pipeline** theory to handle software **pipelined loops** in user programs. More specifically, the problem Theory of Modulo-Scheduled **Pipelines** R. Govindarajan Supercomputer Education and
144.16.67.13/~govind/papers/TR-96-1.ps.gz

Word-size Parallelism - Andrej Brodnik (Correct)
Finally, these operations represent a **single instruction** from the processor's point of view and we differ in the implementation of a parallel FOR **loop** with its functions F i and in the implementation makes the algorithm even more feasible for modern **pipeline** architectures. From the theoretical point of
valjhun.mat.uni-lj.si/~brodnik/Andrej/Papers/WordParallel.ps.gz

Improving Instruction Throughput and Memory Latency Using.. - Tsai, Zheng, Yew (Correct)
parallelism is limited by the size of the **single instruction** window. Multithreaded architectures, such program structures such as doubly-nested WHILE **loops**, **loops** with multiple control blocks in the **loop** The superthreaded architecture supports a thread **pipelined** execution model to increase the overlap among
www.cs.umn.edu/Research/Agassiz/Paper/tsai.two-dim.ps.Z

NIW: A Simple Superscalar Architecture - Crispin Cowan (Correct)
encoding multiple instructions (two) into a **single instruction** word, similar to VLIW architectures, and we of parallelism present in compiled code, barring **loop** unrolling, is in fact two[5, 10]Since sequences are analyzed comparing a conventional **pipelined** RISC processor to one modified as described. 1
ftp.csd.uwo.ca/pub/csd-technical-reports/413/report413.ps.Z

Documents 21 to 40 Previous 20 Next 20

Try your query at: Amazon Barnes & Noble Google (RI) Google (Web) CSB DBLP

Active Page Architectures for Media Processing
(1999) (Make Corrections) (1 citation)
Justin Hensley, Mark Oskin, Diana Keen, Lucian-Vlad Lita and Frederic T....

Abstract: In this paper, we compare the performance of various media applications on vector and VLIW Active Page [OCS98] memory architectures. Active Pages is an intelligent memory system that associates simple functions with each physical page of data. Previous investigations have demonstrated substantial speedups compared to a conventional memory system for media and data-intensive applications. This study explores two processing elements for use in Active Pages: a very long instruction word processor ... (Update)

Context of citations to this paper: More

.... **to the architecture and the Active Pages group is now turning its attention to integration of conventional processors into the memory[52, 94].** Recently, NEC has designed an FPGA with integrated DRAM[86] in a very finegrain architecture. The highlight of this design appears to:..

Cited by: More
Balancing Computation and Memory in High Capacity.. - Perissakis (2000) (Correct)

Similar documents (at the sentence level):
5.0%: Exploiting ILP in Page-Based Intelligent Memory - Mark Oskin Justin (1999) (Correct)

Active bibliography (related documents): More All
1.0: Reducing Cost and Tolerating Defects in Page-based.. - Oskin, Keen.. (Correct)
0.3: A Platform for Prototyping PIMOS System Services - Schermerhorn (Correct)
0.3: ActiveOS: Virtualizing Intelligent Memory - Mark Oskin Frederic (1999) (Correct)

Similar documents based on text: More All
0.2: TriMedia CPU64 Architecture - van Eijndhoven, Sijstermans.. (1999) (Correct)
0.1: Low-Power Design of Page-Based Intelligent Memory - Justin Hensley Aneet (Correct)
0.1: Care and Feeding of High-Performance Processors with.. - Frederic Chong (Correct)

BibTeX entry: (Update)

J. Hensley, M. Oskin, D. Keen, L-V. Lita, and F.T. Chong. "Active Page Architectures for Media Processing". In First Workshop on Media Processors and DSPs, 32nd Annual Symposium on Microarchitecture, November 1999. 181 http://citeseer.nj.nec.com/hensley99active.html More

```
@misc{ hensley99active,
  author = "J. Hensley and M. Oskin and D. Keen and L. Lita and F. Chong",
  title = "Active Page Architectures for Media Processing",
  text = "J. Hensley, M. Oskin, D. Keen, L-V. Lita, and F.T. Chong. Active Page Architect
    for Media Processing. In First Workshop on Media Processors and DSPs, 32nd
    Annual Symposium on Microarchitecture, November 1999. 181",
  year = "1999",
  url = "citeseer.nj.nec.com/hensley99active.html" }
```

Citations (may not include all citations):
75 Machine multicomputer: An architectural evaluation (context) - Noakes, Wallach et al. - 1993
51 Space-time scheduling of instructionlevel parallelism on a R.. - Lee - 1998
42 The case for intelligent RAM: IRAM - Patterson - 1997
35 Processing in memory: the Terasys massively parallel PIM arr.. (context) - Gokhale, Holmes et al. - 1995
35 Active pages: A computation model for intelligent memory - Oskin, Chong et al. - 1998

34 The energy efficiency of IRAM architectures - Fromm - 1997
16 Mapping applications to the rapid configurable architecture - Ebeling - 1997
15 Morph: A system architecture for robust high performance usi.. - Chien, Gupta - 1996
14 Hierarchical processors-and-memory architecture for high per.. - Miled, Eigenmann et al. - 1996
8 Exploiting ilp in page-based intelligent memory - Oskin, Hensley et al. - 1999
8 The Trimedia TM-1 PCI VLIW mediaprocessor (context) - Slavenburg, Rathnam et al. - 1996
5 VLIW processor for multimedia applications (context) - Holmann, Yoshida et al. - 1996
5 Parallel processing RAM chip with 256Mb DRAM and quad proces.. - Murakami, Shirakawa et al. - 1997
4 Design of an active memory system for network applications (context) - Asthana, Cravatts et al. - 1994
3 Activeos: Virtualizing intelligent memory - Oskin, Chong et al. - 1999
2 IEEE Signal Processing Magazine (context) - Seshan, processing - 1998
1 Flexram: An advanced intelligent memory system (context) - Kang, Huang et al. - 1999